**Software Implementation and Testing Document**

**For**

**Group Phaze 5**

Version 2.0

**Authors**:

Faith Miller

Mackenzie Knight

Eduardo Antonini

Tristan Garcia

Katie Rombeiro

# 

# **1.** **Programming Languages (5 points)**

Our project is divided in two parts: backend and frontend. Both parts are written in JavaScript. While the backend is written without any special libraries or APIs, the frontend is written with the React library. We chose JavaScript for several reasons — mainly because of its popularity in the web app developer community, but also because some members of our group had previous experience with the React library. JavaScript also offered the benefits of being an object oriented language, which is the programming approach our group members feel most confident using.

# **2.** **Platforms, APIs, Databases, and other technologies used (5 points)**

We are still using Atom as our primary coding platform. This is where we will run and push our full project once it is completed. We’ve also started using CodePen to test and run our backend game implementation on its own so we could make sure our game runs properly before connecting to our frontend. We are using node package manager (npm) for running the frontend React App.

# **3.** **Execution-based Functional Testing (10 points)**

Execution-based functional testing for the frontend has included adding code in HTML and CSS with React and checking to see what the UI on the localhost looks like. This has allowed those working on the frontend to both know whether or not their code compiles and if the UI looks as intended.

Execution-based functional testing for the backend has included using an online application called CodePen to fix compile errors and run the game implementation coded in JavaScript. We decided that the most efficient way to test our game implementation was to first use prompt for player input and alert for output. Using keyboard input and alerts for output has allowed us to first verify the compilation and functionality of the game implementation before integrating it with the frontend. Now that we know our game implementation is bug free, we can now connect the frontend with the backend with minimal errors.

# **4.** **Execution-based Non-Functional Testing (10 points)**

We’ve performed some small execution-based non-functional testing. By using the inspect aspect of Google Chrome, it allowed us to see if the console.log call actually outputted the desired information and also allowed us to fix any warnings due to a clear warning description. Because we do not focus on energy efficiency, we can focus on writing our functions in an easier manner instead of worrying about functionality and speed. We also have been making sure that the state variables are being updated correctly for the number of players and also the player usernames. In further coding we will be focusing on the non-functional testing of player input for the actual playing of the game.

# 

# **5.** **Non-Execution-based Testing (10 points)**

When coding the game implementation of our project, walkthroughs were performed of the code. For example, when a few of the game implementation functions were completed, those who were working on the game implementation had a Zoom meeting and they discussed the functions and made sure they were correct and had no mistakes. During those meetings, they also inspected each other's code to make sure there were no mistakes that they missed. In addition, those who coded the game implementation played a few games of Phaze 5 through prompts and alerts to make sure that the entire game functioned properly before connecting the game implementation to the user interface. Prompts were used to ask for user input throughout the game and alerts were used to output information to the screen. When the user interface was being coded, those who were working on the front-end had a Zoom meeting and did a walkthrough of each functional requirement completed since the last meeting. In these meetings, they explained what issues arose during the coding of each functional requirement and this allowed for each person to inspect the code and check for any mistakes.